Stir to Pour: Efficient Calibration of Liquid Properties for Pouring Actions

Stir to Pour: Efficient Calibration of Liquid Properties for Pouring Actions

Tatiana Lopez-Guevara[1,2], Rita Pucci[3], Nicholas K. Taylor[2],
Michael U. Gutmann[1], Subramanian Ramamoorthy[1], Kartic Subr[1]

***Abstract*— Humans use simple probing actions to develop intuition about the physical behavior of common objects. Such intuition is particularly useful for adaptive estimation of favorable manipulation strategies of those objects in novel contexts. For example, observing the effect of tilt on a transparent bottle containing an unknown liquid provides clues on how the liquid might be poured. It is desirable to equip general-purpose robotic systems with this capability because it is inevitable that they will encounter novel objects and scenarios. In this paper, we teach a robot to use a simple, specified probing strategy – stirring with a stick– to reduce spillage when pouring unknown liquids. In the probing step, we continuously observe the effects of a real robot stirring a liquid, while simultaneously tuning the parameters to a model (simulator) until the two outputs are in agreement. We obtain optimal simulation parameters, characterizing the unknown liquid, via a Bayesian Optimizer that minimizes the discrepancy between real and simulated outcomes. Then, we optimize the pouring policy conditioning on the optimal simulation parameters determined via stirring. We show that using stirring as a probing strategy result in reduced spillage for three qualitatively different liquids when executed on a UR10 Robot, compared to probing via pouring. Finally, we provide quantitative insights into the reason for stirring being a suitable calibration task for pouring –a step towards automatic discovery of probing strategies.**

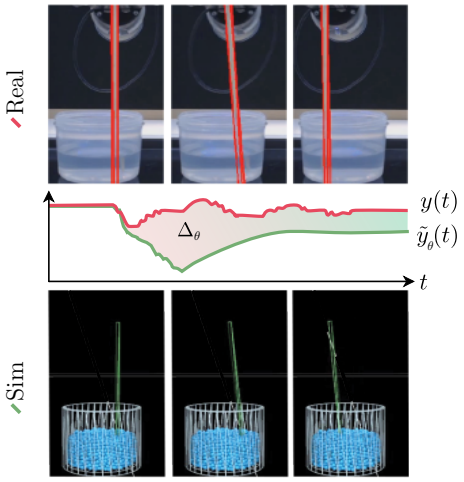

Fig. 1. Synchronized stirring actions with a UR10 Robot (Top) and NVidia Flex simulator (Bottom). Measurements (Middle) consist of the inclination of a stick, that can freely pivot at the gripping point, across time: $y(t)$ (in red) for real and $\tilde{y}(t)$ (in green) for sim. Discrepancy Δ_θ is proportional to the shaded area.

I. INTRODUCTION

The development of general-purpose robots that can learn to manipulate liquids has the potential to impact multiple sectors including engineering, medicine and the service industries. Applying machine learning techniques to learn about unknown fluids is challenging due to several difficulties including sensing methods for generating data, complicated models underpinning flow and the lack of a shape or appearance prior. These hurdles are typically overcome in robotics applications either by using specific parametric approximations [1], such as assuming parabolic trajectories for pouring [2], or by using a fluid simulator as a model [3], [4]. In this paper, we adopt the latter approach since it is more general.

When using simulation, it is necessary to strike a compromise between accurate and efficient (fast) models. For robotics applications that reason about liquids in closed-loop, the latter is often more important [3], [5]. However, fast models are usually approximate and therefore introduce an additional challenge. In addition to simulation input parameters such as the shape of the containers it is also necessary to learn approximate fluid parameters with potentially no mapping to real physical parameters [4]. Work in the field of intuitive physics argue that humans use similar approximations to perform complex tasks [6], [7], [8], [9].

The model mismatch or "reality gap" requires the estimation of input simulation parameters as a precursor to any task-specific optimisation, i.e. if the target task is to pour a liquid optimally, the simulation parameters corresponding to the liquid in question need to be estimated first. Then, these parameters are used to determine the optimal actions to be executed by the robot. Previous work in fluid manipulation has either assumed that these input parameters are known *a priori* [10], [11], [12], [13], [14] or are estimated by executing the target task in a calibration stage [5], [4], [15]. However, it is often an advantage to be able to assess these parameters by performing a simpler task that does not require manual intervention (cleaning and replenishing) and that minimize the risk of damaging the robot.

We propose a method to estimate properties of unknown liquids using stirring as a simple probing strategy. We evaluate the efficacy of our probing strategy by pouring unknown liquids. First, we observe the inclination of a stick that is used by a real robot to stir an unknown liquid. We continuously tune fluid simulation parameters until the inclination of the

[1]University of Edinburgh, 10 Crichton St, Edinburgh EH8 9AB.
[2]Heriot-Watt University, Currie, Edinburgh, EH14 4AS.
[3]University of Udine.
Authors emails in appearing order:
t.l.guevara@ed.ac.uk, rita.pucci@uniud.it, n.k.taylor@hw.ac.uk, michael.gutmann@ed.ac.uk, s.ramamoorthy@ed.ac.uk, k.subr@ed.ac.uk

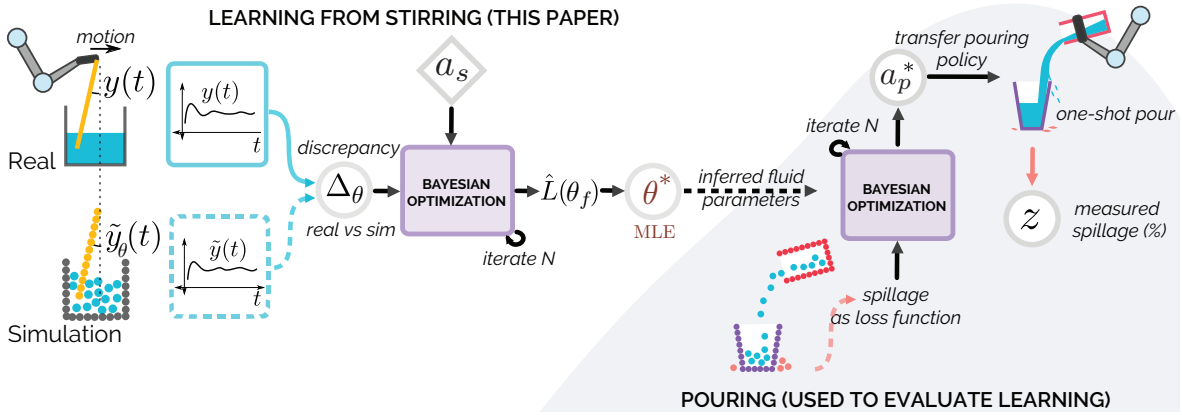

Fig. 2. This paper focuses on learning parameters of liquids by stirring. Learning parameters of liquids θ by stirring with the motion pattern a_s. The discrepancy Δ_θ between the real $y(t)$ and simulated $\tilde{y}(t)$ time signals is obtained in real time. The efficacy of learning is evaluated by executing one-shot pouring and measuring the percentage spillage z.

stick in the simulation matches real observations. The central hypothesis is that these parameters will be useful to then perform a different task –pouring– with the same unknown fluid (Fig. 2). We show that using stirring (as opposed to pouring) as a probing strategy results in reduced spillage for three qualitatively different liquids when executed on a UR10 Robot. In addition to simplifying the inference of parameters, our method also improves the efficacy of the pouring task.

II. RELATED WORK

System Identification for Fluids Dynamics: System identification comprises a set of methods to characterize a dynamical system for control. There is a large body of work on system identification applied to fluid dynamics [16], [17], [18], [19], [20]. Most of these methods obtain a lower dimensional representation of the system that is easier to control, by assuming knowledge of the underlying model [16] or by learning it directly from data [17], [18], [19], [20]. The result is a reduced model that is efficient, but specific to each system or task.

Robots interacting with fluids: A different line of work is focused on general simulation models that can be used across tasks, at the expense of reduced accuracy to keep the computation tractable. For example learning a pouring policy using approximate simulation [10], [11], [5], [4], [13], [14], a combination of simulation and real observations [21], or differentiable simulation [15]. All these methods require previous knowledge about the specific simulation parameters, which are assumed to be known [10], [11], [13], [14] or obtained via calibration with the same pouring task [5], [4], therefore involving human intervention.

Most of the work in this category is focused on pouring water or liquids with similar viscosity [13] to water. To our extent, this is the first work studying the manipulation of liquids with a wider range of viscosity values, such as glycerin and gel.

Estimation of physical fluid properties: There are also methods that don't assume any underlying model at all, and directly estimate real-physical parameters of liquids from data using robots. For example, to obtain the volume [22], height of the body of liquid [23], 3D shape of the container [24] and dynamic viscosity [25], [26], [27]. These methods exploit special measurement equipment such as RGBD cameras, microphones or tactile sensors for parameter estimation. In our context, knowledge of exact physical parameters is not useful since they do not correspond to their counterparts in simulation, except when using a high-fidelity simulation that involves higher computation times [12].

Contributions: The high-level contributions of this paper are that we: (1) propose a simple, online calibration action (stirring) decoupled from the target manipulation task (pouring), (2) show adaptability of the obtained estimates for pouring. We analyze the behaviour of the system in 3 liquids with a wide range of viscosities (water, glycerin and gel).

III. PROBLEM DEFINITION

Let $a_s \in \mathcal{A}_s$ denote actions performed during calibration ("stirring"). Let $\theta \in \Theta$ define the parameters controlling the behaviour of the liquid in the simulation-based model. Let $y(t)$ describe the real observed inclination of the stick while stirring with the robot. Let $\tilde{y}_\theta(t)$ be the simulated observed inclination while stirring in the simulator with parameters θ. We define the observed discrepancy (for stirring) over the duration T of action a_s as

$$\Delta_\theta = \int_T \left(y(t) - \tilde{y}_\theta(t)\right)^2 \, \mathrm{d}t. \tag{1}$$

Let $a_p \in \mathcal{A}_p$ be a pouring action and let z denote the corresponding spillage (as a percentage of the poured liquid) observed when a_p is executed by the robot.

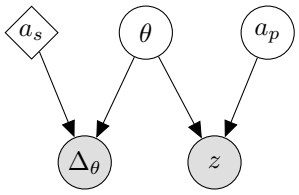

Fig. 3. Graphical model showing relationships between variables: a_s and a_p are stirring and pouring actions; Δ_θ is the discrepancy in real and simulated inclinations; z is the measured relative spillage; and θ is the parameter that characterizes the *shared* property of the simulated liquid. By stirring, we wish to obtain a maximum likelihood estimator for θ^*. During pouring, we use θ^* to determine an optimum a_p^* which reduces z.

Here, we analyze the problem of inferring parameters $\theta^* \in \Theta$ of the liquid, given a space of calibration (stirring) actions $\mathcal{A}_s$ *that are different from* the space of goal (pouring) actions $\mathcal{A}_p$. We quantify the suitability of $\mathcal{A}_s$ by measuring the percentage of liquid spilled z while performing optimized one-shot pouring using $a_p^* \in \mathcal{A}_p$ obtained from θ^* (See Fig 3).

Assumptions: We assume that the configuration of the scene including the geometry of the containers and the initial positions are available, or can be estimated using sensors. Also, we rely on the robot's estimation of its end effector pose, to synchronise simulation with reality.

Inference: Given a specific $\mathcal{A}_s$, say stirring using a given motion pattern, the goal of the inference step is to estimate the best θ^* in simulation such that the discrepancy Δ_{θ^*} is minimal. The optimization consists of $k = 1 : N$ iterations. At each iteration k, an action a_s is executed by the robot (real) and in simulation (sim) using a hypothesized parameter θ. The resulting discrepancy Δ_θ^k, calculated using Eq. 1, together with the parameter θ are provided to a Bayesian Optimizer that learns a regression of Δ_θ over θ using a Gaussian Process [28]:

$$\Delta_\theta^k \sim \mathcal{GP}(\mu^k(\theta), \kappa^k(\theta, \theta^{'})), \tag{2}$$

An approximation of the likelihood [29], [30] can be computed using the cumulative density function (CDF) Φ of the standard Normal distribution and a threshold ϵ as:

$$\hat{L}^N(\theta) \propto \Phi\left(\frac{\epsilon - \mu^N(\theta)}{\sigma^N(\theta)}\right). \tag{3}$$

The goal is to determine the maximum likelihood estimator MLE as:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\, \hat{L}^N(\theta) \tag{4}$$

In addition to the MLE, the likelihood function (Fig. 8-*Left*) can also be used to compute samples from the posterior via a Hamiltonian Monte Carlo technique [31], by first generating samples from the prior distribution $\theta \sim \mathcal{U}(\theta_{min}, \theta_{max})$ and evaluating their likelihood according to Eq. 3.

Evaluation: We quantitatively evaluate the suitability of $\mathcal{A}_s$ for the problem by measuring percentage spillage using an optimized action $a_p^* \in \mathcal{A}_p$. This is due to the lack of an existing ground truth of the parameters in the simulator given its approximate nature. Since our contribution concerns the calibration task, we use a pouring strategy exactly as prescribed by previous work [4]. They use a simulator to identify a_p^*, by defining the loss function to be the ratio of the spilled particles to the total number of particles simulated. The j^{th} iteration of their method therefore involves executing the simulator with action $a_p^j \in \mathcal{A}_s$ and θ^*. The minimization results in a_p^* after a finite number of iterations (15 in our case). Finally, we execute a_p^* using the robot and measure the percentage of liquid spilled.

Algorithm 1: Inference Model (k^{th} iter).

Input: Stirring action a_s
Output: Discrepancy Δ_θ
Sample from prior $\theta^k \sim \mathcal{U}(\theta_{min}, \theta_{max})$
Get sim obs. $\tilde{y}_t(t) = NvFlex(a_s, \theta^k)$
Get real obs. $y(t) = Robot(a_s)$
Compute discrepancy: Δ_θ using (Eq. 1)
Return Δ_θ

IV. EXPERIMENTAL SETUP

For all our experimets, we used a UR10 robot equipped with a gripper in combination with MoveIt [32]. Simulations were executed using NVIDIA Flex [33], running on Dell Alienware Laptop with a NVIDIA GeForce GTX 1070 Graphics card and 8GB of RAM. We used a Kern 2.5k weighting scale to measure the spillage when pouring. We used two Logitech HD Pro C920 webcams to capture orthogonal views from the stick when the robot stirs the liquid. Each stirring iteration takes around 30 seconds and it does not involve human intervention. Each pouring iteration takes between 2 and 4 minutes depending on the liquid, out of which the majority involves manual operation and cleaning. The experiments performed in this paper (including evaluation and comparison with prior work) consumed a total of about 40 robot hours and 30 person hours for supervision. We used the Engine for Likelihood-Free Inference (ELFI) [34] for the calibration process with Algorithm 1. as the model.

A. Robot Setup

Stirring: The robot's gripper is used to hold a stick so that it is free to pivot at the gripping point (see Fig. 1). Before stirring begins, the stick is vertical and partly submerged in the liquid. The motion of the end effector is limited to a plane $\mathcal{P}$ parallel to the ground plane. Due to this motion, and the the resistance encountered by the stick due to the liquid, at any instant t, the stick might deviate from its vertical position to $y(t)$. The inclination is intricately dependent on the velocity of the end effector and the physical properties of the liquid and the stick. $y(t)$ is estimated using a Hough Line detector with OpenCV [35] on the video feed from 2 webcams with image planes orthogonal to $\mathcal{P}$. We average the discrepancy across views to obtain Δ_θ. The stick is wrapped

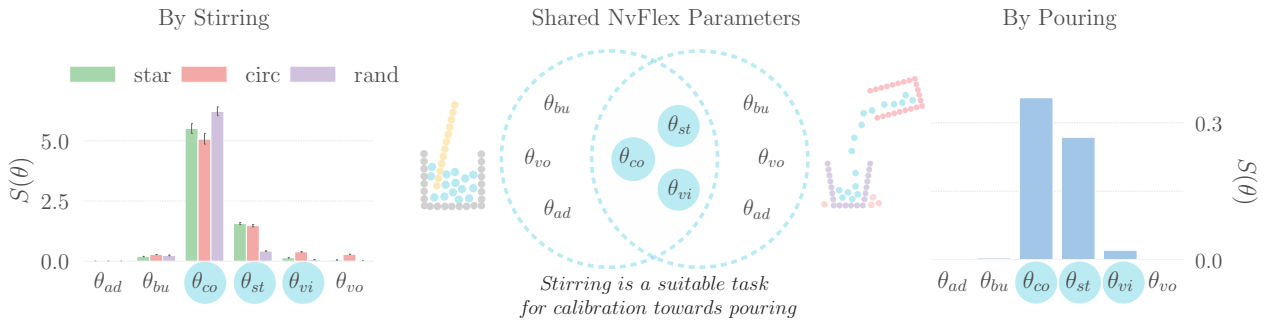

Fig. 4. Model sensitivity analysis for six NvFlex parameters affecting liquid behaviour in simulation for (Left) Stirring using different patterns (star, circular, and random) and (Right) Pouring. We are interested in the shared parameter between tasks (Middle). Parameter names are abbreviated as (ad: Adhesion, bu: Buoyancy, co: Cohesion, st: Surface Tension, vi: Viscosity, vo: Vorticity).

in bright green paper to reduce the error of the estimates of $y(t)$. The light levels were kept similar across experiments to avoid issues with the detection. The position of the end effector of the robot is queried at 30Hz and supplied to the simulator which replicates the executed action. The inclination produced in simulation at instant t is recorded as $\tilde{y}_\theta(t)$. The space of stirring actions $\mathcal{A}_s$ is discrete and determined by the stirring pattern. In this work we used a continuous sequence, $\mathcal{A}_s = \{a_s^i\},\ i = 1, \cdots, m$ that visually follows one of three: $9-$point star, circular or random patterns. Each action a_s^i corresponds to the desired position of the robot's end effector.

Pouring: We replicate the one-shot pouring solution from [4]. For completeness, we review their method here using our notation. The space of pouring actions $\mathcal{A}_p$ is two dimensional and continuous. The 2D space is parameterized by a constant angular velocity and the relative distance between source and target containers $a_p^i = \left(\omega^i, p^i\right)$. After 15 iterations of the optimizer over a_p given θ^*, the robot obtains an estimate for the optimal pouring action a_p^*, which it then executes. We measure the percentage of liquid spilled by the robot over 5 repetitions of the above experiment.

B. Model Sensitivity

We started with the span of six input variables to the simulator as the whole parameter space $\Theta \in \mathcal{R}^6$ of liquids that can be handled by our system. We denote each parameter as $\theta \equiv (\theta_{ad}, \theta_{bu}, \theta_{co}, \theta_{st}, \theta_{vi}, \theta_{vo})$ where the six variables are called (in NVIDIA Flex) adhesion, buoyancy, cohesion, surface tension, viscosity and vorticity confinement (See Table I for more detail). Despite their nomenclature, we observed that these parameters do not behave as their physical (real) namesakes. We believe this is caused by the approximate nature of the simulator.

Stirring: We obtained $j = 10$ jittered samples within the range of each parameter $\theta_p^j \sim \mathcal{U}(\theta_{p,min}, \theta_{p,max})$ uniformly from the lower and upper bounds of the parameter values (See Table I). Keeping all other parameters at their default values, we recorded the inclinations $\tilde{y}_{p,\theta}^j(t)$ observed in simulation with the parameter set to each of the j^{th} jittered values. We computed the sensitivity of the experiment with respect to each parameter as the average variance of the observations with respect to the parameter as

$$S(\theta_p) \equiv \frac{1}{T} \sum_t \mathrm{Var}_{\theta_p} \left[|y_{p,\theta}(t)| \right]. \tag{5}$$

Pouring: We performed exactly the same experiment as above, for stirring, except that we execute the simulator with pouring actions instead of stirring. Instead of defining the variation with respect to the observed inclinations, in this case, we measure the variation in the ratio of spilled particles across multiple repetitions of pouring.

V. RESULTS AND DISCUSSION

Model Sensitivity: We first analyzed the impact of the parameters θ on the stirring experiment (Fig. 4-*Left*). Then, we analyzed their impact on pouring (Fig. 4-*Right*). Most parameters show little or no variation, suggesting that their values do not significantly impact the stirring experiment. The three parameters that do exhibit consistent variation across the patterns are: cohesion, surface tension and viscosity (Fig. 4-*Middle*). We further found that, the sensitivity to the surface tension parameter was caused by instabilities (particles behaving chaotically), rather than because of the action performed. So, we set this parameter to its default value and only consider viscosity and cohesion $\Theta \in \mathcal{R}^2$, which significantly lowers the calibration time.

Confidence of parameter estimates from stirring: We computed a measure of confidence of the parameter estimates using the interquartile range (IQR) on samples from the posterior along the dimensions of each θ. We used the No-U-Turn Sampler algorithm [31] with 4 chains with 1k samples each. The samples were rescaled between 0 and 1 before computing the statistics. Fig. 5 show the IQR for the parameters (cohesion and viscosity), motion patterns (star, circular, random) and liquids (water, glycerin, gel). Of the three motion patterns generated, we found the star pattern to achieve the most confidence in the inferred parameters and therefore is the one we chose to evaluate on the pouring task. We also report the obtained MLE estimates θ^* across $r = 5$ repetitions in Table. II computed using (Eq. 4). These values

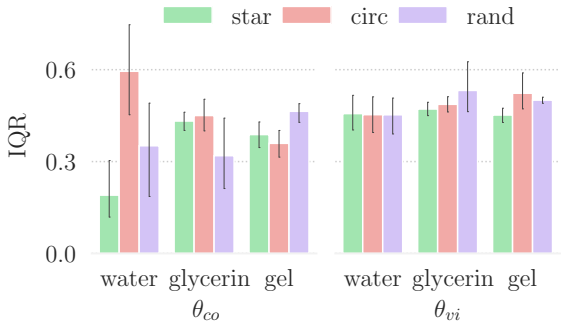

Fig. 5. Measure of confidence for the inferred parameters. We used the Interquartile range (IQR) of samples drawn from the posterior along the dimensions of each θ. Samples from the star pattern produce the lowest IQR (highest confidence) for cohesion θ_{co}. All patterns show similar results for viscosity θ_{vi}.

correspond to the simulation parameters that were used to evaluate pouring in the next 2 sections. The high IQR reported for viscosity reflects the irrelevance of such parameter for estimation. This can also be seen in the likelihood contour of Figure 8 and on the variable MLE estimates along such dimension.

Calibration by stirring vs by pouring: We compared the percentage spillage z achieved by our algorithm which calibrates by stirring against the method proposed in [4] which calibrates by pouring using a training cup. Although it would seem intuitive that applying the same task to train must result in lower spillage under test conditions, our results indicate the contrary. Fig. 6 plots z vs N, where N is the number of iterations of the Bayesian Optimizer (B.O.) used to estimate θ^*. Using our stirring approach, the spillage is less than 5% even with only 10 iterations, under half the corresponding figure when the robot was trained with pouring. At $N = 20$ iterations, our approach almost achieves zeros spillage (which is lower than learning from pouring at $N = 60$ iterations). We believe the difference between pouring and stirring at 60 iterations is caused by randomness in the simulator.

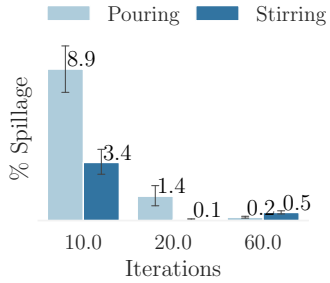

Fig. 6. Effect of two calibration methods in the deployment task measured as the decrease of spillage with respect to the number of iterations.

Pouring other liquids: We observed a similar trend across three different liquids (Fig. 7): as N is increased, the spillage reduces. However, the degree of spillage is significantly higher for glycerin and gel. We empirically noticed that this is due to adhesion effects, making the liquids stick to the pouring

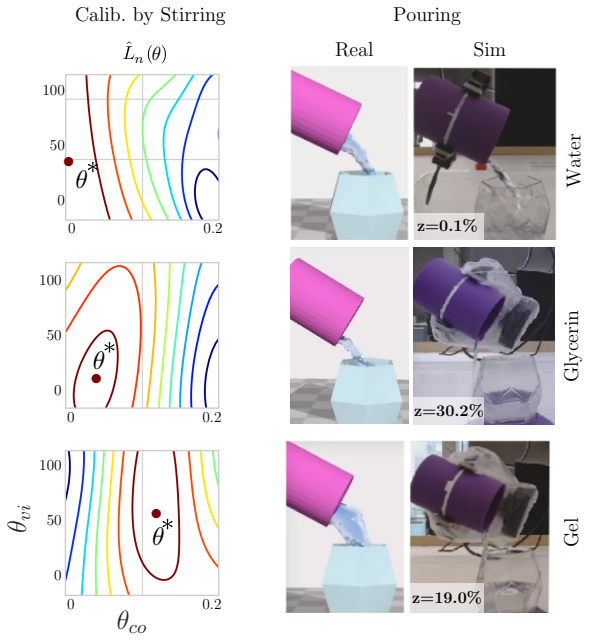

Fig. 8. (Left) Contour plots of the likelihood of the fluid parameters after stirring water, glycerin and gel for one of the repetitions. (Middle) Pouring with the real robot. (Right) Synchronized pouring in simulation.

container in the real world (Fig. 7-*Right*). Unfortunately, the adhesion parameter did not have any effect in simulation, creating a strong model mismatch, both in stirring and pouring. We believe that the choice of the approximate simulator is the source of error during spillage[1]. However, the capability to infer parameters within a limited gamut of expressibility is still a valuable addition to the toolkits proposed by existing methods.

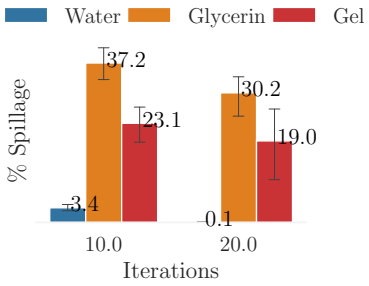

Fig. 7. Effect of the parameters inferred after performing the stirring action 10 and 20 times on three liquids.

Iterations vs repetitions in calibration: We use the term iterations to refer to the number of steps performed by the optimizer. At each iteration, it samples an action, executes a simulation (yielding $\tilde{y}_\theta(t)$), executes the action on the real robot (yielding $y(t)$) and uses Δ_θ as the evaluation of the loss function for that iteration. At the end of N iterations, we have a single estimate for θ^*. Due to the stochasticity inherent to the process, θ^* is a single realization of the optimum inferred

[1]Due to the current COVID-19 situation we were unable to run additional experiments on calibration by pouring to validate this.

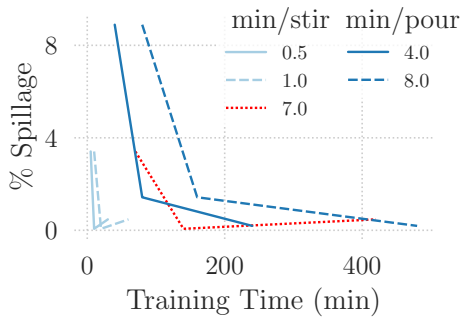

Fig. 9. We compared percentage spillage achieved by our method (calibration by stirring) against calibration by pouring as a function of the time spent in calibration. Stirring is both quicker (solid light-blue curve, 0.5 min per stir) as well as more efficient compared to learning by pouring (solid dark-blue curve, $4min$ per pour). The dashed and dotted curves are hypothetical calibration times. Another advantage of learning by stirring is that it is completely autonomous, since no cleanup is required.

parameters. To robustify this estimate, we repeat the whole experiment by performing N iterations again to yield another estimate of θ^*. We then average these estimates, across *repetitions*, obtain the final θ^*. So performing r repetitions with N iterations each requires rN steps of the Bayesian Optimizer (B.O.), and therefore rN executions of actions by the robot. We recommend $r = 5$ and $N = 20$. Optimizing this combination, so that more robust estimates are obtained for equal calibration effort, is an interesting avenue for future work.

Calibration times: Each of the rN iterations of the optimizer in calibration requires one execution of the calibration task. The stirring method takes about 0.5 minutes. On the other hand, calibrating by pouring [4] takes around 4 minutes per data sample. Even if stirring was only as efficient as pouring in terms of the number of optimization iterations, this already offers an eightfold saving $(8\times)$ in time during estimation. In addition, since stirring is more efficient when smaller number of iterations are used, the savings in calibration time by switching from pouring to stirring is significant. This gap is evident in the plot shown in Fig. 9 which compares the spillage during testing resulting from the two different calibration approaches. The solid curves represent real times taken per calibration task for the two approaches. The dashed curves correspond different hypothesis of doubling the time per iteration for each stir/pour. Even if stirring took 7 mins per stirring (which is heavily exaggerated), the spillage (dashed red curve) is comparable to that achieved by "learning by pouring" (solid blue curve). The plot also shows that the total calibration time can be hundreds of minutes as opposed to calibrating by pouring.

Simultaneously sampling $\mathcal{A}_s$ and $\mathcal{A}_p$: We proposed an algorithm that executes rN actions from $\mathcal{A}_s$, infers θ^* and finally performs pouring. For one-shot pouring, the optimizer samples M actions, executes them in simulation and uses the ratio of spillage in simulation as the loss function to generate an optimized action a_p^* which when executed on the real robot results in a spillage of $z\%$. Again, since we use B.O., just as for calibrating, z is a single realization of a random variable. We obtain a more robust estimate by performing s repetitions (hence the error bars in all plots with spillage on the Y-axis). Thus, for each pouring task on a different liquid, the total number of actions sampled is $rNsM$. One possibility would be to reallocate effort by increasing M while setting $r = 1$. That is, for each repetition of pouring we only use a single repetition of inference. This strategy performs better overall due to the joint sampling of $\mathcal{A}_s$and $\mathcal{A}_p$. If calibration is being performed *solely with the goal of pouring*, we recommend that an a_p^* be estimated for each repetition of inference (θ^*).

VI. CONCLUSIONS AND FUTURE WORK

We have proposed and evaluated a new calibration experiment that decouples the calibration action (stirring) from the final task (pouring) while adapting to liquids with widely different properties. We demonstrated that stirring leads to reduced spillage for water compared to other methods. We presented results for calibrating and adapting the pouring to other liquids. Calibration by stirring is preferable to calibration by pouring because it is easy to automate, it is time efficient and avoids the mess involved due to spillage. To our knowledge, this is the first work studying liquids that range from low to high viscosity. We also discussed the several design decisions involved, along with quantitative justification and recommendations for prospective use-cases.

Our work is limited to studying the effect of a fixed set of stirring actions that were selected after careful analysis in simulation. We believe an interesting avenue for future work lies in studying how such actions can be generated automatically using information-based metrics [36], [37], such as maximizing the information gain after each stir. Another interesting direction is to relax the current assumption of knowledge about the shapes and containers and including uncertainty in the estimation of the stick configuration.

APPENDIX

TABLE I
RANGE OF PARAMETERS ON NVFLEX USED IN EXPERIMENTS.

Parameter θ	Abbrev.	θ_{min}	θ_{max}
Adhesion	ad	0.0	0.1
Buoyancy	bu	0.3	2.0
Cohesion	co	0.0	0.2
Surface Tension	st	0.0	50.0
Viscosity	vi	0.0	120.0
Vorticity Confinement	vo	0.0	120.0

ACKNOWLEDGMENTS

This work is supported by the Engineering and Physical Sciences Research Council (EPSRC), as part of the CDT in Robotics and Autonomous Systems at Heriot-Watt University and The University of Edinburgh. The authors would also like to thank the reviewers for their helpful comments.

TABLE II

MLE ESTIMATORS FOR EACH LIQUID, PARAMETER AND PATTERN. RANGES OF PARAMETERS IN TABLE I WERE RE-SCALED TO [0-1].

Liquid	θ_{co}			θ_{vi}		
	Star	Circ	Rand	Star	Circ	Rand
water	0.03 ± 0.07	0.00 ± 0.00	0.07 ± 0.14	0.54 ± 0.35	0.80 ± 0.41	0.46 ± 0.29
glycerin	0.16 ± 0.14	0.02 ± 0.04	0.04 ± 0.07	0.06 ± 0.09	0.40 ± 0.52	0.88 ± 0.18
gel	0.65 ± 0.16	0.84 ± 0.15	0.64 ± 0.34	0.36 ± 0.27	0.51 ± 0.48	0.75 ± 0.28

REFERENCES

[1] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives," *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 910–922, 2011.

[2] Z. Pan and D. Manocha, "Motion Planning for Fluid Manipulation using Simplified Dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 0, 2016. [Online]. Available: http://arxiv.org/abs/1603.02347

[3] C. Schenck and D. Fox, "Reasoning About Liquids via Closed-Loop Simulation," in *Robotics: Science and Systems (RSS)*, 2017. [Online]. Available: http://arxiv.org/abs/1703.01656

[4] T. Lopez-Guevara, N. K. Taylor, M. U. Gutmann, S. Ramamoorthy, and K. Subr, "Adaptable pouring: Teaching robots not to spill using fast but approximate fluid simulation," in *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, 2017, pp. 77–86. [Online]. Available: http://proceedings.mlr.press/v78/lopez-guevara17a.html

[5] C. Schenck and D. Fox, "Visual Closed-Loop Control for Pouring Liquids," in *International Conference on Experimental Robotics (ICRA)*, 2017. [Online]. Available: http://arxiv.org/abs/1610.02610

[6] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18 327–18 332, 2013.

[7] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman, "Physics 101: Learning physical object properties from unlabeled videos." in *BMVC*, vol. 2, no. 6, 2016, p. 7.

[8] C. J. Bates, I. Yildirim, J. B. Tenenbaum, and P. Battaglia, "Modeling human intuitions about liquid flow with particle-based simulation," *PLoS computational biology*, vol. 15, no. 7, p. e1007210, 2019.

[9] J. J. R. Van Assen, P. Barla, and R. W. Fleming, "Visual features in the perception of liquids," *Current biology*, vol. 28, no. 3, pp. 452–458, 2018.

[10] A. Yamaguchi and C. G. Atkeson, "Differential Dynamic Programming for Graph-Structured Dynamical Systems : Generalization of Pouring Behavior with Different Skills," in *IEEE-RAS International Conference on Humanoid Robots*, no. 2, 2016.

[11] L. Kunze and M. Beetz, "Envisioning the qualitative effects of robot manipulation actions using simulation-based projections," *Artificial Intelligence*, jan 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370214001544

[12] Z. Pan and D. Manocha, "Feedback Motion Planning for Liquid Pouring Using Supervised Learning," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[13] C. Do, C. Gordillo, and W. Burgard, "Learning to pour using deep deterministic policy gradients," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3074–3079.

[14] M. Kennedy, K. Schmeckpeper, D. Thakur, C. Jiang, V. Kumar, and K. Daniilidis, "Autonomous precision pouring from unknown containers," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2317–2324, 2019.

[15] C. Schenck and D. Fox, "Spnets: Differentiable fluid dynamics for deep neural networks," in *CoRL*, 2018.

[16] C. W. Rowley, "Model reduction for fluids, using balanced proper orthogonal decomposition," *International Journal of Bifurcation and Chaos*, vol. 15, no. 03, pp. 997–1013, 2005.

[17] J.-N. Juang and R. S. Pappa, "An eigensystem realization algorithm for modal parameter identification and model reduction," *Journal of guidance, control, and dynamics*, vol. 8, no. 5, pp. 620–627, 1985.

[18] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[19] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (sindyc)," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 710–715, 2016.

[20] ——, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[21] A. Yamaguchi and C. G. Atkeson, "Stereo Vision of Liquid and Particle Flow for Robot Pouring," in *IEEE-RAS International Conference on Humanoid Robots*, no. c, 2016.

[22] C. Do, T. Schubert, and W. Burgard, "A Probabilistic Approach to Liquid Level Detection in Cups Using an RGB-D Camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[23] H. Liang, S. Li, X. Ma, N. Hendrich, T. Gerkmann, and J. Zhang, "Making sense of audio vibration for liquid height estimation in robotic pouring," *arXiv preprint arXiv:1903.00650*, 2019.

[24] S. Brandi, O. Kroemer, and J. Peters, "Generalizing pouring actions between objects using warped parameters," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 616–621.

[25] C. Elbrechter, J. Maycock, R. Haschke, and H. Ritter, "Discriminating liquids using a robotic kitchen assistant," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 703–708.

[26] H. Saal, J.-A. Ting, and S. Vijayakumar, "Active sequential learning with tactile feedback," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 677–684.

[27] T. Takahashi and M. C. Lin, "Video-guided real-to-virtual parameter transfer for viscous fluids," *ACM Trans. Graph.*, vol. 38, pp. 237:1–237:12, 2019.

[28] A. McHutchon and C. E. Rasmussen, "Gaussian process training with input noise," in *Advances in Neural Information Processing Systems*, 2011, pp. 1341–1349.

[29] J. Lintusaari, M. Gutmann, R. Dutta, S. Kaski, and J. Corander, "Fundamentals and recent developments in approximate Bayesian computation," *Systematic Biology*, vol. 66, no. 1, pp. e66–e82, Jan. 2017.

[30] M. Gutmann and J. Corander, "Bayesian optimization for likelihood-free inference of simulator-based statistical models," *Journal of Machine Learning Research*, vol. 17, no. 125, pp. 1–47, 2016.

[31] M. D. Homan and A. Gelman, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1593–1623, Jan. 2014.

[32] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[33] M. Macklin and M. Müller, "Position based fluids," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 104, 2013.

[34] J. Lintusaari, H. Vuollekoski, A. Kangasrääsiö, K. Skytén, M. Järvenpää, M. Gutmann, A. Vehtari, J. Corander, and S. Kaski, "Elfi: Engine for likelihood free inference," 2017.

[35] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[36] S. Kleinegesse and M. U. Gutmann, "Efficient bayesian experimental design for implicit models," in *Proceedings of Machine Learning Research*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, 16–18 Apr 2019, pp. 476–485.

[37] S. Kleinegesse and M. Gutmann, "Bayesian experimental design for implicit models by mutual information neural estimation," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.